CEM · CIRCOLO ESPERANTISTA MILANESE · QUADERNO DI DIVULGAZIONE

Artefarita Inferenco

«Inferenza Artificiale» — capire davvero l'Intelligenza Artificiale

★

Un piccolo libro per il CEM (Circolo Esperantista Milanese): che cosa fa (e cosa non fa) un modello come Claude, con la matematica essenziale, il vocabolario indispensabile e alcuni progetti in lingua internazionale.

di Ruben Conti

EDIZIONE · 2026 · LINGUA: ITALIANO CON TERMINI CHIAVE IN ESPERANTO

PREFAZIONE

Perché un esperantista dovrebbe leggere questo libro

Gli esperantisti hanno una vecchia familiarità con un'idea potente: che una struttura regolare, costruita dall'uomo, possa farci comunicare meglio. L'Intelligenza Artificiale è un'altra di queste strutture — solo che è fatta di numeri.

Questo libro nasce da una convinzione semplice: per usare bene l'AI non serve essere matematici, ma serve *capire cosa accade sotto il cofano*. Le sigle (LLM, MCP, token, prompt) intimidiscono finché restano parole magiche; diventano strumenti nel momento in cui se ne afferra il meccanismo. Il nostro filo conduttore è già nel titolo. Quella che chiamiamo «intelligenza» artificiale, a guardarla da vicino, è soprattutto **inferenza**: deduzione statistica a partire da esempi. Da qui il nome che dà il titolo al volume, *artefarita inferenco*[EO] — inferenza artificiale.

Useremo come esempio pratico e ricorrente **Claude Opus 4.8** di Anthropic, nel contesto in cui dà il meglio — la programmazione

— ma i concetti valgono per qualunque modello. E poiché scriviamo per il **CEM**, chiuderemo con due progetti che intrecciano AI ed Esperanto.

COME LEGGERE QUESTO LIBRO

I termini chiave compaiono anche in Esperanto, *tiel*[EO] (così): passaci sopra il mouse per la traduzione. Le formule sono spiegate a parole subito dopo: nessuna va «risolta», solo capita. Il **Capitolo 2** contiene un laboratorio interattivo: lì il libro si tocca con le mani.

CAPITOLO 1

Artefarita Inferenco: il nome è la tesi

Cominciamo da una piccola provocazione: l'AI non è intelligente. Non nel senso in cui lo siamo noi. Quello che fa è *inferire*.

La parola «intelligenza» ci porta a immaginare comprensione, intenzioni, coscienza. Una rete neurale artificiale — *artefarita neŭra reto*[EO], in inglese *Artificial Neural Network*, ANN — non possiede nulla di tutto ciò. Esegue un'operazione che la statistica conosce da oltre un secolo e che porta un nome preciso: **inferenza**. Inferire significa stimare qualcosa che non si conosce a partire da esempi che si conoscono. Se ho visto mille foto di gatti etichettate «gatto», posso *inferire* che la milleunesima, simile alle altre, sia ancora un gatto. Nessuna comprensione di che cosa *sia* un gatto: solo una stima di probabilità basata su regolarità.

Questo spostamento di prospettiva — da «macchina che pensa» a «macchina che stima» — non sminuisce l'AI. Al contrario, la rende comprensibile. Un modello come Claude prevede **miliardi di volte** la stessa cosa: dato ciò che è venuto finora, qual è la

prosecuzione più probabile? La differenza tra un giocattolo e un sistema che aiuta a scrivere software sta tutta nella *qualità* di quella stima, e nel modo in cui è stata affinata.

> L'IDEA PORTANTE DEL LIBRO
>
> Ogni volta che leggerai «il modello capisce», sostituisci mentalmente «il modello stima la risposta più probabile». Quasi tutto, nelle pagine che seguono — i costi, le allucinazioni, il modo giusto di fare domande — discende da questa sola sostituzione.

Come si affina una stima? Misurando quanto si sbaglia e correggendosi. È il cuore matematico dell'apprendimento, e merita un capitolo a sé — con un laboratorio che potrai muovere con le dita.

CAPITOLO 2

La funzione di perdita e la discesa verso lo zero

Un modello impara cercando il punto in cui sbaglia di meno. Quel «quanto sbaglio» ha un nome — *funzione di perdita* — e una forma: una valle. Imparare significa scendere a fondo valle.

Immagina di voler tracciare la retta che meglio descrive una nuvola di punti (la classica **regressione lineare**). Ogni retta che provi commette un errore: la distanza fra ciò che la retta prevede e ciò che i dati dicono davvero. Sommando questi errori otteniamo un solo numero, la *perdo*[EO] — la perdita, in inglese *loss*. L'obiettivo dell'addestramento è renderla **il più vicina possibile a zero**.

La formula, spiegata pezzo per pezzo

Nella sua forma minima, per un singolo esempio con due caratteristiche in ingresso, la perdita si scrive così:

$$\text{Loss} = (y - (x_1 w_1 + x_2 w_2))^2$$

SIMBOLO	CHE COS'È
y	Il valore *vero*, quello che vorremmo indovinare (la risposta corretta dell'esempio).
x1, x2	Le **caratteristiche** in ingresso (*trajtoj*[EO]): i numeri che descrivono l'esempio.
w1, w2	I **pesi** (*pezoj*[EO]): quanto conta ciascuna caratteristica. Sono gli unici numeri che il modello può cambiare.
x1w1+x2w2	La **previsione** del modello: ogni caratteristica moltiplicata per il suo peso, il tutto sommato.
(y − …)	Il **residuo**: di quanto abbiamo mancato il bersaglio.
$(\dots)^2$	Il quadrato: rende positivo ogni errore (sopra o sotto pari peso) e punisce di più gli sbagli grossi. È anche ciò che dà alla valle la sua forma liscia.

Tradotta in parole: *prendi la tua previsione, confrontala con la verità, eleva al quadrato lo scarto.* Fai la media su tutti gli esempi e hai un unico numero che dice quanto il modello è bravo, oggi, con i pesi

che ha. Cambia i pesi, cambia il numero. La domanda diventa geometrica: **quali pesi rendono la perdita minima?**

Scendere lungo la curva

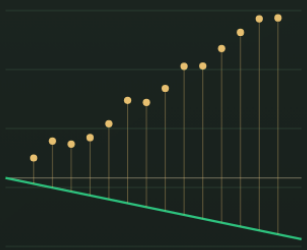
Se mettiamo su un grafico la perdita in funzione di un peso, otteniamo una curva a forma di scodella. In alto sui bordi: pesi pessimi, errore enorme. In fondo: il peso migliore, errore minimo. Il modello parte da un punto a caso e **scende**, un passetto alla volta, nella direzione di massima pendenza verso il basso. Questo metodo si chiama *discesa del gradiente* (***gradienta malsupreniro***[EO]). Il «gradiente» è semplicemente la pendenza: dice da che parte è la discesa e quanto è ripida.

Nel laboratorio qui sotto puoi vederlo accadere. A sinistra la nuvola di punti con la retta che si raddrizza; a destra la stessa storia vista come una pallina che rotola in fondo alla valle della perdita. Premi *Allena* e guarda le due cose muoversi insieme: sono lo stesso fenomeno.

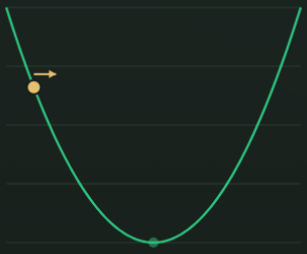
Laboratorio — la discesa del gradiente

RETTA DI REGRESSIONE ⇔ VALLE DELLA PERDITA

DATI & RETTA Ŷ = W·X



PERDITA L(W) — CERCA IL MINIMO



peso w = -0.670 · perdita L = 86.988 · minimo a w* = 1.930 · passo 0

Non una valle, ma molte: i domini

I tre pulsanti del laboratorio cambiano *dominio*: gli stessi calcoli, dati diversi. Nota che la valle cambia forma e il fondo si sposta. È il punto cruciale da visualizzare: **non esiste un'unica curva di perdita**. Ogni compito — tradurre, riconoscere immagini, scrivere codice — vive in un suo paesaggio, con la sua valle e il suo minimo. Un modello reale non scende lungo una curva in due dimensioni, ma lungo una superficie in *miliardi* di dimensioni (un peso per ogni parametro). L'idea, però, è esattamente quella che hai appena toccato con il dito.

PERCHÉ «MILIARDI DI CALCOLI»

Ogni passo di discesa misura la pendenza rispetto a *ciascun* peso e lo aggiorna. Con centinaia di miliardi di pesi e moltissimi passi, si arriva a numeri di operazioni che sfidano l'immaginazione. Qui sta il costo — energetico ed economico — dell'addestramento; ed è anche il motivo per cui i modelli, una volta addestrati, vengono *riusati*: ridiscendere la valle da capo costa troppo.

CAPITOLO 3

1943: McCulloch, Pitts e lo schema FTI

Prima dei computer, due uomini molto diversi immaginarono il neurone come un interruttore logico. Da quell'intuizione discende, in linea retta, tutto ciò che oggi chiamiamo rete neurale.

Nel 1943 il neurofisiologo **Warren McCulloch** e il giovane logico autodidatta **Walter Pitts** pubblicarono un articolo dal titolo austero, *A Logical Calculus of the Ideas Immanent in Nervous Activity*. La loro tesi: un neurone può essere descritto come un'unità che riceve segnali, li somma pesandoli e «scatta» (si attiva) soltanto se la somma supera una soglia. Tutto qui — ma è il primo *modello matematico* del neurone, l'antenato diretto dei pesi `w` che hai visto nel Capitolo 2.

Quel neurone elementare incarna, di fatto, tre momenti che ancora oggi descrivono ogni rete — uno schema che possiamo riassumere con la sigla **FTI**:

FTI	MOMENTO	NEL NEURONE DEL 1943

F	**Feature** — caratteristica	I segnali in ingresso: ciò che il neurone «vede».
T	**Transformation** — trasformazione	La somma pesata degli ingressi e il confronto con la soglia.
I	**Inference** — inferenza	L'uscita: attivo / non attivo. La «decisione» stimata.

È lo stesso ritmo che ritroviamo in Claude e in qualunque modello moderno: prendi dei dati (*feature*), trasformali attraverso strati di pesi (*transformation*), produci una stima (*inference*). Cambiano la scala — da un neurone a centinaia di miliardi — e la matematica della soglia (oggi più morbida e derivabile), non l'impianto concettuale.

NOTA ONESTA SULLA SIGLA

McCulloch e Pitts non usarono la sigla «FTI»: il loro linguaggio era quello della logica e della biologia. *Feature–Transformation–Inference* è una chiave di lettura moderna, utile per riconoscere lo stesso scheletro nei sistemi di oggi. La usiamo come ponte didattico, non come citazione storica. In termini tecnici corretti, i tre momenti si chiamano: **F** — *input* o *estrazione delle caratteristiche* (feature extraction); **T** — la *combinazione lineare pesata* seguita da una *funzione di attivazione* (l'unità del 1943 è una «unità a soglia», in inglese *threshold logic unit*); **I** — il *passaggio in avanti* (forward pass) o *inferenza* in senso stretto, cioè l'uso del modello già addestrato per produrre una previsione.

CAPITOLO 4

Le radici statistiche: Galton e Pearson

La «retta che si raddrizza» del Capitolo 2 non l'ha inventata l'informatica. Ha quasi centocinquant'anni, e due padri vittoriani.

Se l'AI è inferenza, allora i suoi veri nonni sono gli statistici. **Sir Francis Galton**, nell'Inghilterra di fine Ottocento, studiando come la statura dei figli «tornasse verso la media» rispetto a quella dei genitori, coniò il termine destinato a una fortuna immensa: *regressione*. Quella linea che riassume una nuvola di punti — la retta di regressione — nasce lì.

Il suo allievo **Karl Pearson** diede alla cosa solide fondamenta matematiche. A lui dobbiamo il *coefficiente di correlazione* (la misura, fra −1 e +1, di quanto due grandezze si muovono insieme) e buona parte dell'apparato della statistica moderna; fondò perfino il primo dipartimento universitario di statistica. Galton intuì, Pearson formalizzò.

IL FILO CHE LEGA TUTTO

La retta di regressione di Galton e Pearson *è* il modello del Capitolo 2; la «perdita» che minimizziamo è la somma degli scarti al quadrato, un'idea ottocentesca (il metodo dei minimi quadrati). Quando una rete neurale «impara», sta facendo, su scala mostruosa, ciò che Galton faceva con carta e matita: tirare la linea che meglio si adatta ai dati. L'AI non ha rotto con la statistica: l'ha portata in un'altra dimensione.

Tenere a mente questa genealogia è il miglior antidoto alla mitologia: dietro la parola luccicante «intelligenza artificiale» c'è una parentela rispettabile e antica di medie, scarti e correlazioni.

CAPITOLO 5

Da dove vengono le allucinazioni

Un modello che «inventa» un dato falso con tono sicuro non si è rotto: sta facendo esattamente il suo mestiere. Capirne il perché ci vaccina contro le delusioni.

Abbiamo detto che un modello stima la prosecuzione più probabile. La parola chiave è *probabile*, non *vera*. Quando Claude genera una frase, sta campionando da una distribuzione appresa: sceglie parole che, nei suoi dati, seguivano bene parole simili. Quasi sempre «plausibile» coincide con «corretto». Ma quando i dati erano scarsi, contraddittori o assenti, il modello produce comunque qualcosa di fluente — e quel qualcosa può essere semplicemente falso. Questo è ciò che chiamiamo **allucinazione** (*halucino*[EO]).

Due conseguenze importanti. La prima: l'allucinazione non è una bugia, perché non c'è intenzione di ingannare — non c'è *nessuno* dietro a volerlo. È un effetto collaterale dell'inferenza priva di un ancoraggio alla realtà. La seconda: si riduce dando al modello

contesto vero al momento giusto. È precisamente il mestiere del protocollo MCP (Capitolo 6): collegare il modello a fonti reali — documenti, database, strumenti — così che non debba indovinare ciò che può leggere.

L'AI NON È SENZIENTE

Nessuna coscienza, nessun desiderio, nessuna comprensione vissuta: un modello è una funzione matematica, per quanto colossale, che trasforma numeri in numeri. Lo diciamo senza sminimizzare la sua utilità. Anzi: proprio perché non «sa» nulla nel senso umano, sta a noi fornirgli i fatti e verificarne le uscite. La fiducia si concede agli strumenti, non si presta loro come a una persona. La novità più recente di Opus 4.8, non a caso, è proprio una maggiore *onestà*: tende a segnalare quando non è sicuro, invece di affermare con sicurezza cose non supportate.

CAPITOLO 6

Il vocabolario dell'AI

Cinque parole bastano a muoversi con sicurezza: *chatbot e agenti*, *token*, *prompt e skills*, *MCP*, *maieutica*. Le vediamo una per una.

6.1 Chatbot e agenti

Un **chatbot** è una finestra di dialogo: tu scrivi, il modello risponde, turno dopo turno. È conversazione. Un **agente** (*agento*[EO]) è qualcosa di più: a un agente non fai domande, gli *assegni un lavoro*. L'agente pianifica i passi, usa strumenti (apre file, esegue codice, fa ricerche), verifica i risultati e torna con il compito svolto. La stessa «mente» statistica, due modi d'uso: rispondere oppure agire.

Con Opus 4.8 questa seconda modalità ha fatto un salto. In **Claude Code**, lo strumento per programmatori, una funzione chiamata *Dynamic Workflows* permette al modello di pianificare un compito complesso e poi **delegarlo a centinaia di sotto-agenti in parallelo**, verificando le uscite prima di consegnarle. Anthropic la descrive come capace di portare a termine

migrazioni di codice su *centinaia di migliaia di righe*, dall'inizio fino all'integrazione finale. L'agente, qui, è un piccolo cantiere coordinato.

6.2 Token: la moneta del modello

I modelli non leggono lettere né parole intere, ma **token**: frammenti di testo (una parola breve è spesso un token; una lunga si spezza in due o tre). Tutto si misura in token, e — aspetto pratico — tutto si *paga* in token. Si distinguono:

TIPO	CHE COS'È	ESEMPIO
Input	I token che *mandi* al modello: la tua domanda, i documenti allegati, il contesto.	la tua richiesta + i file
Output	I token che il modello *genera* in risposta.	la risposta scritta

L'output costa di più dell'input, perché generare è più oneroso che leggere. Per Claude Opus 4.8 il listino di riferimento (uso via interfaccia di programmazione) è di circa **5 dollari per milione di token in ingresso** e **25 dollari per milione in uscita**; esistono forti sconti riusando contesto già visto (*prompt caching*, fino al 90%) o elaborando a lotti (*batch*, 50%). Il modello dispone inoltre di una «memoria di lavoro» molto ampia — una **finestra di**

contesto fino a un milione di token — entro cui può tenere insieme interi progetti.

TOKEN E PIANI D'USO, IN PRATICA

Chi usa Claude dall'app paga in genere un *abbonamento* con dei limiti d'uso, non i singoli token; chi costruisce soluzioni (come faremo con MCP) usa l'interfaccia di programmazione e paga a consumo, a token. Regola d'oro per contenere i costi: **manda solo il contesto che serve** e chiedi **risposte della lunghezza giusta**. Allegare mezzo archivio «per sicurezza» gonfia l'input; chiedere «rispondi in tre righe» sgonfia l'output.

6.3 Prompt e Skills

Un **prompt** (*instigo*[EO]) è la richiesta che rivolgi al modello. Sembra banale, ma la qualità della risposta dipende moltissimo dalla qualità della domanda. Tre abitudini fanno la differenza: **essere specifici** (dire chi sei, cosa vuoi, in che formato), **dare esempi** di ciò che consideri buono o cattivo, e **chiedere al modello di ragionare a passi** quando il problema è complesso.

Le **Skills** (*lertoj*[EO] — «abilità») sono il livello successivo: pacchetti di istruzioni e conoscenze, scritti una volta e riusati sempre. Invece di rispiegare ogni volta le regole della tua attività, le metti in una Skill e il modello le applica da solo quando servono. Puoi chiedere a Claude stesso di *prepararti* una Skill su misura: gli descrivi il tuo lavoro — le convenzioni, i formati, gli errori da evitare — e lui redige il documento che userà come guida nelle sessioni future. È il modo più semplice per trasformare un assistente generico in uno specialista del *tuo* mestiere.

6.4 MCP: il vero motore

Arriviamo al cuore. **MCP** — *Model Context Protocol* — è uno standard aperto introdotto da **Anthropic nel novembre 2024**. La sua idea è semplice e profonda: dare ai modelli un modo

universale di collegarsi a dati e strumenti esterni. Lo si descrive spesso come una «presa USB-C per l'AI»: prima, ogni collegamento fra un modello e una sorgente richiedeva un adattatore su misura (il problema «N×M»); con MCP, ciascuno si conforma *una volta* allo standard e tutto diventa interoperabile («N+M»). È un'architettura client–server: il modello è il client, i tuoi dati e strumenti vivono dietro dei *server MCP*.

Per chi sviluppa — ed è il mio caso — MCP è ciò che rende possibile costruire **soluzioni verticali**: si sfrutta la potenza di ragionamento di Claude, ma la si *guida* dentro un quadro di regole proprio (un *framework*), così che il modello lavori dentro i binari della nostra applicazione invece di improvvisare. Non è più «l'AI che fa cose a caso»: è l'AI che esegue il *nostro* processo.

UN ESEMPIO REALE: DTOMES4-MCP

Nel mio gestionale industriale **DtoMes4** ho costruito un server MCP, `dtomes4-mcp`, che espone a Claude gli strumenti del progetto: può generare moduli software completi rispettando le convenzioni del framework, ispezionare le maschere, e — punto chiave per noi — **produrre documentazione** e archiviarla nel posto giusto. Claude non «sa» il mio gestionale: glielo faccio leggere e manipolare attraverso MCP, dentro regole che ho scritto io.

La documentazione generata segue **due rami**, con due scopi diversi:

RAMO	PER CHI	DOVE VIVE	COME CI ARRIVA
MLPivot (knowledge base)	Lo sviluppatore e Claude stesso: appunti tecnici, «skills», verbali di sessione.	Un archivio locale con ricerca full-text (database SQLite + FTS).	Lo strumento MCP `archive_document` salva e indicizza il testo.
MLPP (aiuto utente)	L'utente finale dentro l'applicazione: manuali e guide contestuali.	Un database PostgreSQL servito da un visualizzatore in-app.	Il testo passa da una «inbox», viene importato e diventa consultabile.

La cosa elegante è che gli stessi contenuti, una volta prodotti, alimentano un prodotto a sé stante: **MLPivot Pro**, un'applicazione autonoma per la gestione documentale in contesti professionali. Combina la ricerca classica per parole (full-text) con la **ricerca vettoriale** (*vektora serĈo*[EO]): invece di cercare le parole esatte, cerca per *significato*, traducendo testi e domande in vettori numerici (gli *embedding*) e confrontandone la

vicinanza. È il modo in cui un archivio smette di essere un cassetto e diventa qualcosa che «capisce» cosa stai cercando — sempre nel senso statistico che ormai conosciamo bene.

6.5 Maieutica: come conversare per produrre davvero

L'ultima parola non è tecnica ma socratica. **Maieutica** è l'arte di far emergere, con buone domande, ciò che è già in potenza. Con un modello funziona sorprendentemente bene, perché la «conversazione» non è chiacchiera: è il modo di costruire, passo dopo passo, il contesto giusto.

In pratica significa lavorare per *iterazioni* invece di pretendere il capolavoro al primo colpo. Si parte da una richiesta chiara, si guarda la prima bozza, si corregge il tiro («più breve», «con un esempio», «tono più formale»), si chiede al modello di mostrare il ragionamento dove conta, e — importante — gli si dice quando sbaglia. Il modello non si offende e non si stanca: ogni turno aggiunge contesto e restringe la valle di possibilità verso la risposta che vuoi. Chi tratta Claude come un oracolo da interrogare una volta ottiene poco; chi lo tratta come un *collaboratore* con cui ragionare a voce alta ottiene molto.

TRE MOSSE MAIEUTICHE

1. Dai il ruolo e l'obiettivo («Sei un revisore di bilanci; trova le incongruenze»). **2.** Chiedi di ragionare prima di concludere, sui problemi difficili. **3.** Reagisci alla bozza invece di ricominciare: è lì che nasce la qualità.

CAPITOLO 7

ML, DL e LLM

Tre sigle che spesso si confondono. In realtà sono cerchi concentrici: il più piccolo dentro il più grande.

Machine Learning (**ML**) — apprendimento automatico — è l'insieme più ampio: qualunque metodo che impara dai dati invece di seguire regole scritte a mano. Ne fanno parte la regressione lineare del Capitolo 2, gli alberi di decisione, e cento altre tecniche, molte delle quali non hanno nulla di «neurale».

Deep Learning (**DL**) — apprendimento profondo — è un sottoinsieme del ML: usa reti neurali con *molti strati* (da cui «profondo»). Più strati significa più trasformazioni in sequenza, e quindi la capacità di cogliere regolarità molto astratte. I grandi modelli linguistici, Claude compreso, sono DL.

Arriviamo così a Claude. Un **LLM** — *Large Language Model*, grande modello linguistico — è un caso particolare di Deep Learning: una rete neurale profonda addestrata su enormi quantità di testo con un compito tanto semplice quanto fecondo, **prevedere il token successivo**. È esattamente l'inferenza del

Capitolo 1, ripetuta miliardi di volte: dato ciò che è venuto finora, qual è la prosecuzione più probabile? Da questo unico meccanismo emergono, come effetto della scala, le capacità che ci sorprendono: scrivere, tradurre, riassumere, programmare. Claude è un LLM.

Una precisazione utile, perché spesso si fa confusione: gli LLM più recenti sono anche *multimodali*, cioè oltre al testo trattano immagini e audio. Il trucco è sorprendentemente unitario: ogni modalità viene convertita nello stesso linguaggio interno fatto di vettori numerici. Un'immagine diventa una sequenza di numeri proprio come un testo; da quel punto in poi il modello non distingue più «parola» da «pixel», lavora su vettori e basta. È per questo che oggi puoi mostrare a Claude una foto e chiedergli di descriverla, o passargli un PDF e farti spiegare un grafico: dietro le quinte, tutto è ridotto alla stessa valuta.

SIGLA	COSA INCLUDE	ESEMPIO TIPICO
ML	Tutto l'apprendimento dai dati	Stimare il prezzo di una casa
DL	ML con reti neurali profonde	Riconoscere il parlato
LLM	Modello DL addestrato sul linguaggio (prevede il token successivo)	Claude: scrivere, tradurre, programmare

CAPITOLO 8

Come si insegna a un modello

Esistono tre grandi pedagogie per le macchine. Cambiano a seconda di *che cosa* il modello riceve insieme agli esempi.

La prima è l'apprendimento **supervisionato** (*kontrolata lernado*[EO]). Diamo al modello coppie domanda–risposta già corrette: migliaia di immagini etichettate «gatto» o «cane», oppure i punti del nostro Capitolo 2 con il loro valore vero. Il modello cerca i pesi che fanno coincidere le sue previsioni con le etichette. È come uno studente che studia su un libro con le soluzioni a fianco.

La seconda è l'apprendimento **non supervisionato** (*nekontrolata lernado*[EO]). Qui non ci sono etichette: diamo solo i dati e chiediamo al modello di *trovare struttura* da solo — raggruppare clienti simili, scoprire temi ricorrenti, costruire quegli *embedding* che abbiamo incontrato nella ricerca vettoriale. È lo studente che, senza soluzioni, impara a riconoscere da sé le famiglie di problemi.

La terza è l'apprendimento **per rinforzo** (*plifortiga lernado*[EO]). Niente risposte pronte, ma un *premio* o una *penalità* dopo ogni tentativo: come si addestra un cane, o come un programma impara a giocare a scacchi vincendo e perdendo migliaia di partite. È centrale per rifinire i modelli linguistici: dopo l'addestramento di base, persone (e altri modelli) valutano le risposte, e quel giudizio diventa il premio che orienta il comportamento — più utile, più onesto, meno dannoso.

METODO	COSA RICEVE	ESEMPIO PRATICO
Supervisionato	Esempi con la risposta giusta	Filtro antispam addestrato su email già classificate
Non supervisionato	Solo dati, nessuna etichetta	Segmentare i clienti in gruppi affini
Per rinforzo	Premi e penalità	Affinare le risposte di un assistente sulla base di giudizi

CAPITOLO 9

AGI e IA generativa: due cose diverse

Sono i due termini più abusati del momento. Distinguerli evita molti fraintendimenti — e qualche delusione.

L'**IA generativa** è ciò che abbiamo in mano oggi: sistemi che *generano* contenuti — testo, immagini, codice, audio — campionando dalle distribuzioni che hanno appreso. Claude è IA generativa. È potentissima entro i compiti per cui è stata addestrata, ma resta *specializzata*: non ha obiettivi propri né una comprensione del mondo che vada oltre le regolarità dei dati.

L'**AGI** — *Artificial General Intelligence*, intelligenza artificiale generale — è invece un'ipotesi: un sistema capace di apprendere e ragionare *attraverso qualunque dominio* al pari (o oltre) un essere umano, trasferendo competenze da un campo all'altro con la flessibilità che noi diamo per scontata. **Non esiste**. È un orizzonte di ricerca e di dibattito, non un prodotto sullo scaffale.

LA CONFUSIONE DA EVITARE

Un modello generativo molto bravo può *sembrare* generale, perché conversa di tutto. Ma «parlare di tutto» non è «capire e padroneggiare tutto»: è ancora inferenza statistica, vasta ma circoscritta. Tenere separate le due idee — ciò che c'è (generativo) e ciò che si ipotizza (AGI) — è segno di alfabetizzazione, non di scetticismo.

CAPITOLO 10

Etica: l'approccio di Anthropic

Se un modello è uno strumento potente e privo di coscienza, la responsabilità di renderlo sicuro è interamente nostra. È la premessa con cui nasce Anthropic.

Anthropic è un'azienda di ricerca fondata attorno a un'idea netta: costruire sistemi di AI *utili, onesti e innocui*, mettendo la sicurezza prima della corsa. Tre scelte concrete lo rendono visibile. La prima è la **Constitutional AI**: invece di affidare ogni giudizio a valutatori umani, si dà al modello una «costituzione» — un insieme esplicito di principi — e lo si addestra a criticare e correggere le proprie risposte alla luce di quei principi. I valori non sono nascosti nel codice: sono scritti, discutibili, migliorabili.

La seconda è la **politica di scalata responsabile** (*Responsible Scaling Policy*): man mano che i modelli diventano più capaci, crescono in parallelo le misure di sicurezza richieste prima di rilasciarli. Ne abbiamo un esempio vivo proprio in questi mesi: il

modello sperimentale più avanzato, chiamato *Mythos*, è stato **trattenuto** dalla diffusione generale per via delle sue capacità in ambito cybersicurezza, in attesa di salvaguardie adeguate. La terza è l'**onestà** come obiettivo misurabile: la novità più sbandierata di Opus 4.8 non è la velocità, ma il fatto che ammetta più spesso i propri dubbi e affermi meno cose non verificate.

PERCHÉ INTERESSA A UN'ASSOCIAZIONE

Per una realtà non-profit, scegliere strumenti di AI è anche una scelta di valori: come vengono trattati i dati, se c'è pubblicità nascosta, quanto è trasparente il comportamento. I prodotti Claude, ad esempio, non ospitano pubblicità e non si lasciano «comprare» raccomandazioni dentro le conversazioni. Non è un dettaglio di marketing: è la differenza fra uno strumento che lavora per te e uno che lavora su di te.

CAPITOLO 11

Claude al lavoro

Il nostro esempio pratico, Claude Opus 4.8, dà il meglio nella programmazione. Ma lo stesso motore serve molti altri mestieri.

La programmazione, in dettaglio

Anthropic descrive Opus 4.8 come il suo modello più capace tra quelli disponibili al pubblico, «alla frontiera» su programmazione, compiti agentici e lavoro di conoscenza. Nel codice ciò significa cose molto concrete: **pianifica con cura**, **regge progetti lunghi** dentro basi di codice grandi, e — dettaglio prezioso — **si accorge dei propri errori**, tanto che un ingegnere esperto può delegargli i compiti più difficili con ragionevole fiducia. Sui banchi di prova di settore (il *benchmark* SWE-bench Pro, che misura la soluzione di problemi reali di software) è passato da circa 64 a circa 69 su cento rispetto alla versione precedente; ma chi lo costruisce insiste più sull'*onestà* che sul punteggio: sbaglia meno e, quando dubita, lo dice.

Il salto vero, però, è *agentico*. Con **Claude Code** e la funzione *Dynamic Workflows*, il modello può affrontare migrazioni che attraversano centinaia di migliaia di righe, scomponendole fra molti sotto-agenti e verificando i risultati rispetto ai test esistenti. È il passaggio dal «suggeritore di righe» al «collaboratore di progetto».

Oltre il codice

Lo stesso modello, fuori dalla programmazione, lavora altrettanto bene su molti terreni:

SETTORE	COSA PUÒ FARE	AVVERTENZA
Scrittura & documenti	Relazioni, presentazioni, fogli di calcolo, sintesi di testi lunghi.	Verifica sempre numeri e citazioni.
Disegno & design	Bozzetti di interfacce, illustrazioni vettoriali, iterazione visiva su una tela digitale.	Non riproduce opere o marchi protetti.

Medicina	Spiegare referti in parole semplici, riassumere letteratura, fare da supporto al ragionamento clinico.	Non è un medico: nessuna diagnosi, supporto e non sostituto.
Ricerca & analisi	Confrontare fonti, strutturare un'indagine, analizzare dati.	Le fonti vanno controllate; può allucinare.

Il filo comune: Claude è eccellente come *amplificatore* di una competenza umana, rischioso come *sostituto* di un giudizio umano. Il valore non è «l'AI fa al posto mio», ma «l'AI mi fa arrivare prima e meglio, e io decido».

CAPITOLO 12

Il panorama: altre iniziative

«AI» non è un'unica cosa. Conviene distinguere tre famiglie: chi costruisce i modelli, chi fornisce la cassetta degli attrezzi, e chi gestisce l'intera filiera. Solo la prima è fatta di *veri* LLM.

Un'osservazione utile da cui partire — il fatto che alcuni prodotti «integrano» l'AI senza essere veri modelli linguistici — è esattamente la chiave per leggere questo panorama. Mettiamo ordine.

Chi costruisce i modelli (i «veri LLM»)

OpenAI è un laboratorio di ricerca e prodotto: addestra grandi modelli (la famiglia GPT) e li offre via app e interfaccia di programmazione. È un concorrente diretto di Anthropic: qui il prodotto *è* il modello. Nella stessa famiglia stanno Anthropic (Claude) e pochi altri. Quando usi questi servizi, stai usando un LLM in presa diretta.

Chi fornisce la cassetta degli attrezzi

TensorFlow (di Google, dal 2015) non è un modello: è una *libreria* open source con cui i ricercatori costruiscono e addestrano reti neurali proprie. È il tornio, non l'oggetto tornito. Chi adopera TensorFlow fabbrica modelli; non «parla» con un'AI già pronta.

Chi gestisce l'intera filiera (le piattaforme ML / MLOps)

Qui sta la maggior parte della lista, ed è la categoria più fraintesa. Queste piattaforme *non sono* LLM: orchestrano il *ciclo di vita* di un modello — raccogliere i dati, addestrare, valutare, mettere in produzione, monitorare. Sono l'equivalente, per il machine learning, di ciò che il «DevOps» è per il software.

INIZIATIVA	DI CHI	COSA PROPONE
Michelangelo	Uber (2017)	Piattaforma interna «ML come servizio»: democratizza il machine learning in azienda, dall'addestramento al monitoraggio. Resa celebre dal suo *feature store*, con oltre diecimila caratteristiche riusabili.

FBLearner Flow	Meta / Facebook (2016)	La «spina dorsale» AI interna: pipeline riusabili che permettono anche a non specialisti di addestrare e mandare in produzione modelli su vasta scala.
Bighead	Airbnb	Piattaforma ML end-to-end pensata per rendere coerente e ripetibile il lavoro dei data scientist interni (con il proprio feature store, Zipline).
SageMaker	Amazon (AWS)	La versione *commerciale* della stessa idea: un servizio cloud che chiunque può affittare per costruire, addestrare e servire modelli senza gestire l'infrastruttura.
Databricks	Databricks Inc.	Piattaforma dati e analitica nata dai creatori di Apache Spark; con strumenti come MLflow unifica dati e ciclo di vita dei modelli (il «lakehouse»).

LA MORALE PER CHI SCEGLIE

Se vuoi *parlare* con un'AI o farle svolgere compiti, ti serve un modello (Claude, GPT…), eventualmente guidato con MCP. Se vuoi *fabbricare e gestire* modelli tuoi, ti servono una libreria (TensorFlow) e una piattaforma (SageMaker, Databricks…). Confondere le due cose è l'errore più comune — e il più costoso. Per quasi tutte le esigenze di un'associazione, la prima strada (un modello pronto, ben guidato) è quella giusta.

CAPITOLO 13

Esperanto e AI

Chiudiamo dove avevamo aperto: con la lingua internazionale. Due progetti mostrano come l'inferenza artificiale possa mettersi al servizio dell'*Internacia lingvo*[EO].

13.1 Il robotino che parla Esperanto, scrive e disegna

L'idea è semplice e felice: un piccolo robot che dialoga in Esperanto, sa scrivere ciò che gli si chiede e sa anche disegnare. Sotto il cofano non c'è nulla di magico, ma proprio gli ingredienti di questo libro. Un modello linguistico (Capitolo 7), oggi multimodale, si occupa di comprendere e produrre lingua; la sintesi vocale dà voce all'uscita e il riconoscimento vocale raccoglie l'ingresso; un modulo di disegno traduce le richieste in tratti o immagini. Il tutto tenuto insieme — ed è il punto — da un *framework* guidato via MCP, perché il robot non improvvisi ma segua un comportamento previsto.

Il valore per il movimento esperantista è doppio. Da un lato, un oggetto concreto e amichevole che parla la lingua: un ottimo ambasciatore nelle scuole e alle fiere. Dall'altro, un banco di prova reale per la qualità dell'Esperanto generato — che ci porta diritti al secondo progetto, perché un robot che parla bene ha bisogno di una grammatica solida da cui inferire.

13.2 Convertire la grammatica di Bertilo Wennergren in «skills»

I modelli conoscono l'Esperanto, ma in modo statistico: l'hanno «visto» nei dati, non l'hanno *studiato*. Per una lingua pianificata, regolare e ben documentata come la nostra, questo è un peccato — perché esiste una fonte autorevole e sistematica: la **PMEG**, la *Plena Manlibro de Esperanta Gramatiko* di **Bertilo Wennergren**.

Il progetto consiste nel **convertire quella grammatica in Skills** (Capitolo 6.3): riorganizzare le regole — participi, accusativo, formazione delle parole, correlativi — in pacchetti strutturati che il modello consulta e applica *prima* di rispondere. L'effetto è di trasformare un'inferenza «a orecchio» in un'inferenza *vincolata* da una grammatica esplicita: il modello continua a stimare la frase più probabile, ma dentro i binari delle regole della PMEG. È lo stesso movimento del mio `dtomes4-mcp`, dove Claude lavora «guidato» dentro un framework — qui il framework è la grammatica della lingua.

UN CIRCOLO VIRTUOSO

Le due iniziative si alimentano a vicenda: la PMEG-in-Skills migliora la qualità dell'Esperanto del robot; il robot, usato dal vivo, rivela dove la grammatica codificata va affinata. È la maieutica del Capitolo 6.5 applicata a un progetto di comunità: si parte da una bozza, la si mette alla prova, la si corregge. Esattamente come si scende a fondo valle — un passo per volta.

CAPITOLO 14

Un framework per la vita associativa

Mettiamo insieme tutti i fili. Quello che abbiamo descritto non è teoria: è il modo in cui l'intero processo amministrativo di un'associazione può essere ricostruito attorno a un'unica struttura coerente.

Il framework **DtoMes4** di cui ho parlato nel Capitolo 6 non è nato in un giorno. È il frutto di circa **diciotto mesi** di costruzione continua, poggiati su **quarant'anni di programmazione**. La vera scoperta di questo percorso non è stata tecnica, ma di metodo: aver trovato la *giusta combinazione* fra l'esperienza maturata in decenni e la programmazione *guidata dall'AI*. Da soli, né l'una né l'altra bastano. L'esperienza senza l'AI è lenta; l'AI senza esperienza è rapida ma cieca, e produce quegli errori plausibili di cui abbiamo parlato. Insieme — l'uomo che conosce il dominio e detta le regole, il modello che esegue dentro quei binari via MCP — il ritmo cambia del tutto.

La conseguenza pratica è che lo **stesso impianto** può reggere l'amministrazione di un'associazione come il CEM: l'anagrafica dei soci e il rinnovo delle quote, il protocollo e l'archivio dei documenti, l'organizzazione di eventi e congressi, la comunicazione verso i soci, la rendicontazione. Non una collezione di programmi scollegati, ma *moduli* di un solo framework, che condividono dati e convenzioni — e che si documentano da soli, perché la generazione della documentazione (MLPivot per chi sviluppa, MLPP per l'utente finale) è parte del processo, non un'aggiunta finale.

L'IDEA, IN UNA FRASE

Un'associazione non ha bisogno di «comprare l'AI»: ha bisogno di un framework costruito sul *suo* modo di lavorare, dentro cui l'AI fa la parte che le riesce meglio — scrivere codice, generare documenti, automatizzare il ripetitivo — mentre le persone mantengono le decisioni. È la differenza fra subire uno strumento e costruirsi uno strumento su misura.

APENDICO

Glossario EO–IT

I termini chiave incontrati nel libro, in Esperanto e in italiano.

ESPERANTO	ITALIANO	INGLESE
artefarita inferenco	inferenza artificiale	artificial inference
artefarita neŭra reto	rete neurale artificiale	artificial neural network
perdo	perdita	loss
gradienta malsupreniro	discesa del gradiente	gradient descent
trajtoj	caratteristiche	features
pezoj	pesi	weights
halucino	allucinazione	hallucination
agento	agente	agent

instigo	richiesta / prompt	prompt
lertoj	abilità / skills	skills
vektora serĈo	ricerca vettoriale	vector search
kontrolata lernado	apprendimento supervisionato	supervised learning
nekontrolata lernado	apprendimento non supervisionato	unsupervised learning
plifortiga lernado	apprendimento per rinforzo	reinforcement learning
Internacia lingvo	lingua internazionale	international language